Shortest Path Planning Algorithm – A Particle Swarm Optimization (PSO) Approach

Patience I. Adamu, *IAENG, Member,* Joshua T. Jegede, Hilary I. Okagbue and Pelumi E. Oguntunde

***Abstract*— Path planning for a mobile robot is a difficult task and has been widely studied in robotics. The objective of recent researches is not just to find feasible paths but to find paths that are optimal with respect to distance covered and safety of the robot. Techniques based on optimization have been proposed to solve this problem but some of them used techniques that may converge to local minimum. In this paper, we present a global path planning algorithm for a mobile robot in a known environment with static obstacles. This algorithm finds the optimal path with respect to distance covered. It uses particle swarm optimization (PSO) technique for convergence to global minimum and a customized algorithm which generates the coordinates of the search space. Our customized algorithm generates the coordinates of the search space and passes the result to the PSO algorithm which then uses the coordinate values to determine the optimal path from start to finish. We perform our experiments using four different environments with population size 100 each *in* a 10 x 10 grid terrain and our results are favorable.**

***Index Terms*— Robotics, Motion Planning, Optimization, Particle Swarm Optimization.**

I. Introduction

DURING the last century, automation has become an extremely fast growing phenomenon, impacting almost all facets of life. Therefore, autonomously navigating robots have become increasingly important (Farritor and Dubowsky [5] and are required in many fields (Willeke and Kunz [15]). Motion planning is one of the important tasks in intelligent control of an autonomous mobile robot (Fogel, [6]). It involves the planning of a collision-free path for a mobile robot as it moves from an initial position to a final position in an environment with obstacles. This problem finds application not only in robotics, but in medicine, virtual reality (Lien, [8]) and bioinformatics (Song and Amato [11]) to mention a few.

Motion planning algorithms finds sequence of valid configurations from the free space to form a path, which the mobile robot takes while avoiding collisions. Finding these configurations deterministically becomes a difficult task as the dimensions of the configuration space increases (Reif [10]). Recently though, variants of optimization based methods have been proposed to solve this problem but some of them used techniques that may converge to local minimum. Examples are of Zhang et al. [16], Deng et al. ([4]), Kim and Lee [7] and Barraquand and Latombe [1] which can be seen especially for complex constraints and different degrees of freedom.

Hence, we present an algorithm that uses particle swarm optimization (PSO) technique as the base optimization algorithm and a customized algorithm which generates the coordinates of the search space. PSO is a stochastic global optimization technique which is population based and inspired by group behaviors in animals.

Recently PSO technique has been applied for optimal pose selection in movement of robotic arm (Wang et al. [13]), detumble and control of space robot (Wang et al. [14]), reducing friction during robotic machining (Chen and Zhang [3]) and their references therein.

Essentially, our path-planning algorithm is used to find a feasible path around an obstacle. Assuming there are no obstacles in the navigation area, the shortest path between the start point and the end point is a straight line (Fig.1). The robot proceeds along this path until an obstacle is detected. At this point, our path-planning algorithm is used to find a feasible path around the obstacle. After avoiding the obstacle, the robot continues to navigate towards the end-point along a straight line until the robot detects another obstacle or the desired destination is reached. The search space is viewed as a grid which can be described by the Cartesian plane. In order to avoid ambiguous solutions, we assume that the robot moves along the mid-points of the cells from one cell to another. Ordinarily, Particle Swarm Optimization can be used to determine the optimal path between the start point and finish point of the robot motion. But, this can only be possible if the coordinates of the search space are known. Hence, our customized algorithm generates 100 coordinates of the search space and passes the result to the PSO algorithm which then uses the best 10 coordinate values to determine the optimal path from start position to the final position.

We use four environments to perform our experiments in a 10 x 10 grid terrain: without obstacle (Figures 1(a-b)), with one obstacle (Figures 2(a-c)), with two obstacles (Figures 3(a-c)) and with three obstacles (Figures 4(a-c)). In each of the environments we calculated the distance covered.

The results show that the optimal distance is approximately the diagonal of the 10 x10 grid in all the environments (Figures 2c, 3c and 4c). This confirms the mathematical assertion that, the shortest distance between two non-adjacent vertices in a quadrilateral is its diagonal.

Manuscript received February 27, 2018; revised March 27, 2018. This work was sponsored by Covenant University, Ota, Nigeria.

P. I. Adamu, H. I. Okagbue and P. E. Oguntunde are with the Department of Mathematics, Covenant University, Ota, Nigeria.
patience.adamu@covenantuniversity.edu.ng
hilary.okagbue@covenantuniversity.edu.ng
pelumi.oguntunde@covenantuniversity.edu.ng
J.T. Jegede is with the Department of Electrical/Electronic Engineering, University of Maiduguri, Maiduguri, Nigeria.

The contribution of this paper is the introduction of a global optimization technique to find the optimal path of a robot in a known environment.

II. Preliminaries And Related Area

A Distance Metrics

A distance metric is a function, $\partial(s,t) \to R,$ which calculates the Euclidean distance between two configurations $s = (s_1, s_2, ...s_n)$ and $t = (t_1, t_2, ...t_n)$ in the Euclidean space.

Mathematically,

$$\partial(s,t) = \sqrt{(t_1 - s_1)^2 + (t_2 - s_2)^2 + ... + (t_n - s_n)^2}$$

This work uses this metric to calculate the distances covered by the mobile robot from the initial position to the end position in the different environments shown in Fig. 1 (environment without any obstacle) and Figs. 2b, 3b and 4b (environments with one (two) (three) obstacle(s) after algorithm has been used to get the configurations). The essence of doing this is to be able to compare the distances covered in Figs. 2b, 3b and 4b with Fig. 1 separately and to ascertain to what extent our algorithm is able to minimize the distance covered in environments with obstacles.

B. Local and Optimal Points of a Function

Some functions have "hills and valleys", where they get to maximum or minimum (optimal) value. It may not be for the whole function but for a particular interval. That is local optimal point. The point that is optimal for the whole function is a global optimal point. There is only one global maximum (and one global minimum) but there can be more than one local maximum or minimum. The function $\cos 3\pi x / x$ in Fig.5 has its global maximum at point (0.1, 5.9) local maximum at point (0.6, 1.35), global minimum at point (0.3, -3.2) local minimum at point (1.0, -1.0).

C. Global and Local Path-Planning

Global path planning requires the environment to be completely known and the terrain should be static. In this approach the algorithm generates a complete path from the start point to the destination point before the robots starts motion. On the other hand, local path planning means that path planning is done while the robot is moving; in other words, the algorithm is capable of producing a new path in response to environmental changes. Assuming that there are no obstacles in the navigation area, the shortest path between the start point and the end point is a straight line between the points. The robot will proceed along this path until an obstacle is detected. At this point, our path-planning algorithm is utilized to find a feasible path around the obstacle. After avoiding the obstacle, the robot continues to navigate towards the end-point along a straight line until the robot detects another obstacle or the desired destination is reached.

D. Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is a stochastic global optimization method based on population. It is inspired by group behaviors in wildlife. It is an optimization technique which provides an evolutionary based search. The term PSO refers to a relatively new family of algorithms that may be used to find optimal or near to optimal solutions to numerical and qualitative problems. It is implemented easily in most of the programming languages since the core of the program can be written in a single line of code and has proven both very effective and quick when applied to a diverse set of optimization problems. PSO algorithms are especially useful for parameter optimization in continuous, multi-dimensional search spaces. PSO is mainly inspired by social behavior patterns of organisms that live and interact within large groups. In particular, PSO incorporates swarming behaviors observed in flocks of birds, schools of fish, or swarms of bees.

E. Particle Swarm Optimization Algorithm

Set iteration counter $i = 0$

- Initialize the parameters ω, c_1 and c_2
- Initialize N random particles p_1, p_2 ... p_N (also called positions) and their velocities v_1, v_2, ... v_N. The velocities indicate the amount of change that is applied to a current position (i.e. particle or solution) to arrive at the updated particle (position). The subscripts indicate the particle number in the swarm.
- Evaluate the fitness of each particle from the objective function $f_n^i = F(p_n)$, where $F(\cdot)$ is the objective function to be optimized.
- Update $\overline{f}_n^i$ and $\overline{p}_n^i$ pair as in (1), where $\overline{f}_n^i$ is the *pbest* and $\overline{p}_n^i$ the pbest-yielding particle in the *i-th* generation.

$$\left[\overline{p}_n^i \;\; \overline{f}_n^i\right] = \begin{Bmatrix} [\overline{p}_n^{i-1} \;\; \overline{f}_n^{i-1}] & \text{if } \overline{f}_n^{i-1} \text{ is better than } \overline{f}_n^i \\ [\overline{p}_n^i \;\; \overline{f}_n^i] & \text{if } \overline{f}_n^i \text{ is better than } \overline{f}_n^{i-1} \end{Bmatrix} \quad (1)$$

That is, compare the current and the previous *pbest* values and retain whichever is better; also retain the corresponding position (or particle) that yielded the pbest.

- Update the global best *gbest* with best fitness $\overline{f}_g^i$. The particle that yields gbest is called the global best particle (or position) $\overline{p}_g^i$. The pair can be obtained from (2)

$$\left[\overline{p}_g^i \;\; \overline{f}_g^i\right] = \text{best}\left([\overline{p}_1^i \;\; \overline{f}_1^i],\ [\overline{p}_2^i \;\; \overline{f}_2^i],\ \ldots,\ [\overline{p}_n^i \;\; \overline{f}_n^i]\right) \quad (2)$$

- Update velocities and positions of each particle according to (6) and (7)

$$v_n^{i+1} = \omega \cdot v_n^i + c_1 \cdot r_1^i \cdot (\overline{p}_n^i - p_n^i) + c_2 \cdot r_2^i \cdot (\overline{p}_g^i - p_n^i)$$

ISBN: 978-988-14047-9-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

(3)

$$p_n^{i+1} = p_n^i + v_n^{i+1} \quad (4)$$

For the PSO implemented in this paper, the global best particle $\bar{p}_g^i$ is slightly perturbed to explore positions in its vicinity using (5). This guarantees faster convergence (Bergh [2]) and reduces the chances of the algorithm getting stuck in a local minimum (or local maxima).

$$v_g^{i+1} = \bar{p}_g^i - p_g^i + \alpha \cdot v_g^i + \beta \cdot r_3^i \quad (5)$$

– Set $i = i + 1$ (6)

– Terminate on convergence (ε is the convergence measure) or when the iteration limit is reached.
– Go to the fourth step.

In (3), r_1, r_2 and r_3 are unit random numbers, c_1 and c_2 are scaling coefficients such that $0 < c_1, c_2 \leq 2$ (Paquet and Engelbrecht [9]), α and β are constants while ω is an inertia weight which may be adjusted dynamically to control the fineness of the search at different stages of the iteration process (Venu and Ganesh [12]). Availability of an expert input in step three increases the convergence speed of the algorithm.

III. Path-Planning Process

The search space is viewed as a grid which can be described by the Cartesian plane. This search space contains small square-shaped cells whose reference point is at the center. Hence, the coordinate of each cell can be described with the x and y points on the Cartesian plane.

In order to avoid ambiguous solutions, we assume that the robot moves along the mid-points of the cells from one cell to another, Also, we assume that the obstacles are placed along the optimal path of the robot motion, that is, the path the robot will take if the search space in free of obstacles (Figure 1(a-c)).

The starting point is (0.5, 0.5) and the finishing point is (9.5,9.5) for a 10 x 10 search space or the starting point is (0.5,0.5) and the finishing point is (99.5,99.5) for 100 x 100 search space. The algorithm is developed in such a way as to handle any square shaped search space.

The algorithm we developed using PSO as the base optimization algorithm requires that a valid number, for example 10 for 10 x 10 or 100 for 100 x 100 search space is specified along with a valid integer 1, 2 or 3 for the number of obstacles to be introduced. Thereafter, the algorithm requires that we specify the coordinates of the obstacles corresponding to the number of obstacles specified earlier. Once this is done, the algorithm generates the coordinates of the search space and then uses PSO to determine the optimal path taking into consideration the obstacles introduced earlier.

Ordinarily, Particle Swarm Optimization can be used to determine the optimal path between the start point and finish point of the robot motion. But, this can only be possible if the coordinates of the search space are known. Practically, it is easier to know the coordinate of the obstacles than the coordinates of the search space with obstacles introduced. Hence, our customized algorithm generates the coordinates of the search space and passes the result to the PSO algorithm which then uses the coordinate values to determine the optimal path from start to finish.

IV. Experiments

Experimental Set up

We use our algorithm in four different experiments: a 10 x10 grid environment without any obstacle Fig. 1a, with one obstacle at point (3.5, 3.5) (Fig. 2a), with two obstacles at points (3.5, 3.5) and (5.5, 5.5) (Fig. 3a) and three obstacles at points (3.5, 3.5), (5.5, 5.5) and (6.5, 6.5) (Fig. 4a). We use the algorithm: which generates the Cartesian coordinates for different population sizes 100, 50, 20 and 10 in the 10 x10 grid environment (Table 1 – Table 3), and then uses PSO to determine the optimal path from the start point to the finish point. Using the distance metric, we calculate the distance covered and compare if it is the minimum distance in all the environments.

Results and Discussions

The results shown in Table 1 – Table 3 are the Cartesian coordinates generated by the algorithm. The coordinate points of population size 100 in Table 1 gave the graphs of Figs. 2(a-c), the coordinate points of population size 100 in Table 2 gave the graphs of Figs. 3(a-c) and the coordinate points of population size 100 in Table 3 gave the graphs of Figs. 4(a-c).

A. *Experiment 1: Environment without Any Obstacle:*

We first of all run our algorithm in an enviroment without obstacle to know the optimal path if there is no obstacle in that enviroment (Fig, 1).

Fig. 1a: Shows the optimal path without obstacle

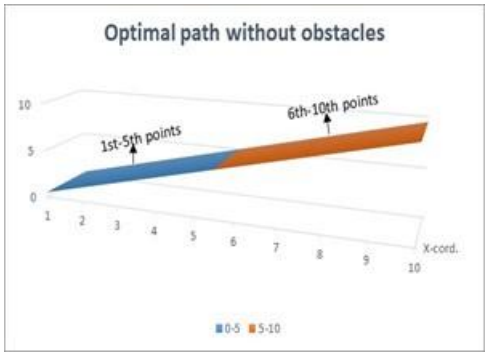

Fig. 1b: Shows the optimal path of Fig. 1a in 3D

ISBN: 978-988-14047-9-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Length of Path:
Using the Euclidean distance metric, we have

$$\partial(start, end) = \sqrt{(9.5-5.0)^2 + (9.5-5)^2} = 12.73$$

This is approximately the diagonal of this environment and mathematically is the shortest distance from the start point to the end point in the diagram. In a 3D environment (Fig. 1b), this can be seen clearly.

B. *Experiment 2: Environment with One Obstacle:*

We use Table 1 to draw the graphs in Figs, 2(a-c). Fig. 2a shows the enviroment with the obstacle positioned at point (3.5, 3.5). Fig. 2c is the 3D representation.

Table 1: One Obstacle at point (3.5, 3.5)

Population Size-100		
Points	x_i-cords.	y_i-cords.
1	0.5021	0.4988
2	1.5025	1.5016
3	2.5009	2.4999
4	4.4976	2.4991
5	4.4972	4.5001
6	5.5006	5.4996
7	6.5005	6.5004
8	7.5003	7.4986
9	8.5066	8.4994
10	9.4980	9.4994

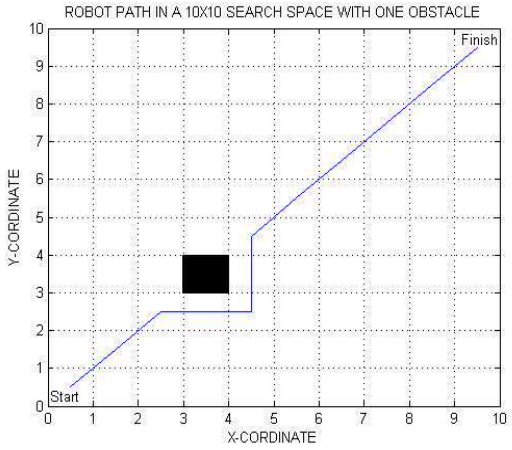

Fig. 2a: shows the path with one obstacle

Fig. 2b: Shows the optimal path (without showing the obstacle) in 2D

Fig. 2c: Shows the optimal path in 3D.

Length of Path:
The distance covered when the population size is 100:

$$\partial(start, end) = \sqrt{(2.5009-0.5021)^2 + (2.4999-0.4988)^2} + (4.4976-2.5009) + (4.5001-2.4991) + \sqrt{(9.4980-4.4972)^2 + (9.4994-4.5001)^2} = 13.87 \text{ units}$$

We see that after encountering the obstacle at point (3.5, 3.5) the algorithm comes back to the optimal path. This can be seen clearly in Fig. 2c. The percentage difference between the distance covered here and the optimal path in no obstacle environment is 0.0114 %. This is negligible; hence, this is the minimum path in this environment

C. *Experiment 2: Environment with Two Obstacles:*

We run our algorithm in an enviroment with two obstacles. Figs. 3a, 3b, and 3c corresponds to the optimal path of Table 2 the population size of 100 both in 2D and 3D. Fig 3a shows the position of the obstacles.

ISBN: 978-988-14047-9-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Table 2: two Obstacles at points (3.5, 3.5) and (5.5, 5.5)

Population Size-100		
Points	x_i-cords.	y_i-cords.
1	0.4996	0.5009
2	1.5003	1.5034
3	2.4999	2.4992
4	4.5008	2.5004
5	4.5014	4.5003
6	6.5005	4.5031
7	6.5007	6.4963
8	7.4996	7.5034
9	8.4995	8.4977
10	9.4990	9.4993

Fig. 3a: shows the optimal path with two obstacles.

Fig. 3b: Shows the optimal path of Fig. 3a.

Fig. 3c: Shows the optimal path of Fig. 3a in 3D.

Path Distance
The distance covered when the population size is 100 (Fig. 4):

$$\partial(start, end) = \sqrt{(2.4999 - 0.4996)^2 + (2.4992 - 0.5009)^2}$$
$$+(4.5008 - 2.4999) + (4.5003 - 2.5004)$$
$$+(6.5005 - 4.5014) + (6.4963 - 4.5031)$$
$$+\sqrt{(9.4990 - 6.5007)^2 + (9.4993 - 6.4963)^2}$$
$$= 15.06$$

Here, again the algorithm forces itself back to the optimal path after encountering the two obstacles. Approximately, the path is the diagonal of the figure. This can be clearly seen in Fig. 3c. Hence it is the optimal path.

D. Experiment 3: Environment with Three Obstacles:
We run our algorithm in an enviroment with three obstacles. Figs. 4a, 4b, and 4c corresponds to the charts of Table 3 the population size of 100. Fig 4a shows the position of the obstacles, Fig. 4b shows the optimal path from start to finish configurations. Fig. 4c is the extension to 3D.

Table 3: Environment with three Obstacles at point (3.5, 3.5), (5.5, 5.5) and (6.5, 6.5)

Population Size-100		
Points	x_i-cords.	y_i-cords.
1	0.4993	0.5008
2	1.5002	1.5030
3	2.5031	2.4970
4	4.5024	2.5024
5	4.5003	4.4972
6	6.5010	4.5028
7	7.5037	5.499
8	7.5010	7.5012
9	8.4984	8.5004
10	9.4990	9.5022

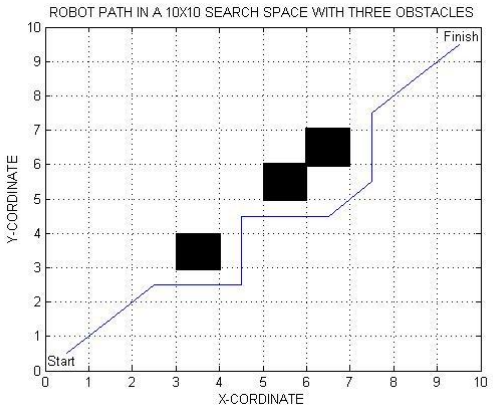

Fig. 4a: shows the possible path with three obstacles.

Fig. 4b: Shows the optimal of Fig. 4a in 2D

ISBN: 978-988-14047-9-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

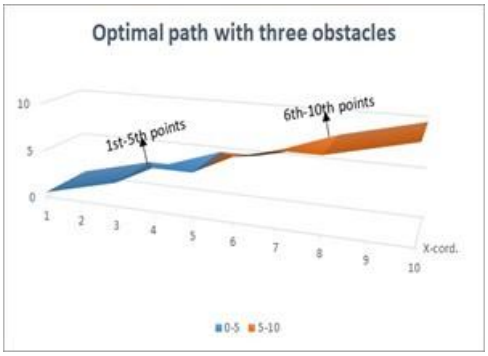

Fig. 4c: Shows the optimal path of Fig. 4a in 3D.

Path Distance:
The distance covered when the population size is 100

$$\partial(sttart, end) = \sqrt{(2.5031-0.4993)^2+(2.4970-0.5008)^2}$$
$$+(4.5024-2.5031)+(4.4972-2.5024)$$
$$+(6.5010-4.5003)$$
$$+\sqrt{(7.5037-6.5010)^2+(5.4990-4.5028)^2}$$
$$+(7.5012-5.4990)$$
$$+\sqrt{(9.4990-7.5010)^2+(9.5022-7.5012)^2}$$
$$=15.07$$

Clearly, this path is optimal in this environment.

Discussions:
In Figs. 2a, 3a, and 4a, it shows that the PSO navigated below the obstacles to get the optimal paths in the different environments. The other path the robot could take, is navigating above the obstacles. Mathematically, the distance covered in both routes are the same. Hence, these paths are the optimal paths in their environments. If the PSO had navigated above the obstacles the path will still be optimal because mathematically the distance covered in both routes is the same. As much as possible we see that the Algorithm try to make the optimal path the diagonal of the 10 x 10 grid environment which is right mathematically. This can be seen, clearly in the 3D environment (Figs. 1c, 2c, 3c and 4c).

V. Concluding Remarks

In this paper, we present a new optimization- based algorithm to find the shortest global path for a robot in a known environment. The algorithm uses Particle Swarm Optimization approach to avoid convergence into a local minimum. With different experiments we show that the algorithm finds the shortest path in any known environment.

VI. Further Work

In this work, we used the algorithm first on an environment without obstacles to get the optimal path. Then we now put the obstacles in the optimal path. In each of the environments: with one obstacle, two obstacles and three obstacles separately, the algorithm try as much as possible to come back to the optimal path if it is not encountering any obstacle. So, our future work will be to put the obstacles scattered in the environments to see how the algorithm gets the optimal path.

Acknowledgment

This research benefited from sponsorship from the Statistics sub-cluster of the *Industrial Mathematics Research Group* (TIMREG) of Covenant University and *Centre for Research, Innovation and Discovery* (CUCRID), Covenant University, Ota, Nigeria.

References

[1] J. Barraquand, and J. C. Latombe, A Monte-Carlo algorithm for path planning with many degrees of freedom. In Robotics and Automation, 1990. Proceedings., IEEE International Conference on pp. 1712-1717, 1990.
[2] F. Bergh van den, "An Analysis of Particle Swarm Optimizers". PhD Thesis. Department of Computer Science, University of Pretoria, pp 15 – 30, 2002.
[3] S. Chen and T. Zhang, (2018). Force control approaches research for robotic machining based on particle swarm optimization and adaptive iteration algorithms. Industrial Robot: An International Journal, 45(1), 141-151.
[4] L. Deng, X. Ma, J. Gu, Y. Li, Z. Xu and Y. Wang, Artificial immune network-based multi-robot formation path planning with obstacle avoidance. International Journal of Robotics and Automation, 31(3), 233-242, 2016.
[5] S. Farritor, and S. Dubowsky, "A Genetic Algorithm Based Navigation and Planning Methodology for Planetary Robot Exploration", Proceeding of the 7th American Nuclear Society Conference on Robotics and Remote Systems, Augusta, GA, 1997.
[6] D. B. Fogel, "What is evolutionary computation?" , IEEE Spectrum, pp. 26-32, 2000
[7] J. J. Kim and J.J. Lee, Trajectory optimization with particle swarm optimization for manipulator motion planning. IEEE Transactions on Industrial Informatics, 11(3), 620-631, 2015.
[8] J.-M. Lien, O. B. Bayazit, R.-T. Sowell, S. Rodriguez, and N. M. Amato. Shepherding behaviors. In Proc. IEEE Int. Conf. Robot. Autom. (ICRA), pp. 4159-4164, 2004.
[9] U. Paquet, and A. P. Engelbrecht," Training Support Vector Machines with Particle Swarms". In Proceedings of International Joint Conference on Neural Networks (IJCNN) Conference, pp 1593 – 1598, 2003.
[10] J. H. Reif. "Complexity of the mover's problem and generalizations". In Proc. IEEE Symp. Foundations of Computer Science (FOCS), pp. 421–427, 1979.
[11] G. Song and N. M. Amato. Using motion planning to study protein folding pathways. In Proc. Int. Conf. Comput. Molecular Biology, pp. 287–296, 2001.
[12] G. G. Venu, and K. V. Ganesh, "Evolving Digital Circuits Using Particle Swarm", In Proceedings oInternational Joint Conference on Neural Networks, pp. 468-471, 2003.
[13] W. Wang, H. Song, Z. Yan, L. Sun and Z. Du, A universal index and an improved PSO algorithm for optimal pose selection in kinematic calibration of a novel surgical robot. Robotics and Computer-Integrated Manufacturing, 50, pp. 90-101, 2018.
[14] M. Wang, J. Luo, J. Yuan and U.Walter, Detumbling strategy and coordination control of kinematically redundant space robot after capturing a tumbling target. Nonlinear Dynamics, Article in press, 2018.
[15] T. Willeke, C. Kunz and I. Nourbakhsh, The Personal Rover Project: The Comprehensive Design of a Domestic Personal Robot, Robotics and Autonomous Systems, Elsevier Science, pp.245-258, 2003.
[16] Y. Zhang, C. Chen and Q. Liu, Mobile Robot Path Planning Using Ant Colony Algorithm. International Journal of Control and Automation, 9(9), 19-28, 2016.